Lazy Evaluation and Subsumption Caching for Search-Based Integrated Task and Motion Planning

Christian Dornhege, Andreas Hertle, Bernhard Nebel

{dornhege,hertle,nebel}@informatik.uni-freiburg.de

Foundations of Artificial Intelligence

University of Freiburg

SPATIAL COGNITION

Planning and Robotics

- Robot skills map to symbolic actions
- Task planning well suited for solving such problems
- Real-world contains
  - Numerical values
  - Arbitrary/unexpected situations
  - Geometric problems

SPATIAL COGNITION

Integrated Task and Motion Planning

- Consider geometric constraints in addition to logical ones
- Predicate semantics defined by external modules
  - Conditions
  - Numerical effects
  - Cost
- Problem: Time intensive (in comparison to logical reasoning)

SPATIAL COGNITION

Generality vs. Efficiency

- Generic interface abstracts modules from planner
- State of the art handling of symbolic planning part
- Agnostic about (slow) module calls
  (besides sorting in conditions)

→ Make the planner aware without loosing soundness or interfering with general priniciples

SPATIAL COGNITION

Techniques

- Caching methods
  - Full State Caching
  - Partial State Caching
  - Subsumption Caching
  - Global Caching
- Lazy module evaluation

SPATIAL COGNITION

Full State Caching

- Cache computation results for all modules as one request
- Cache key pair s, o
- Same s, o requests for
  - Different kind of module – condition or effect for same operator
  - Same module – e.g. revisiting search path from better state
- Easy to implement (generic in planner)

SPATIAL COGNITION

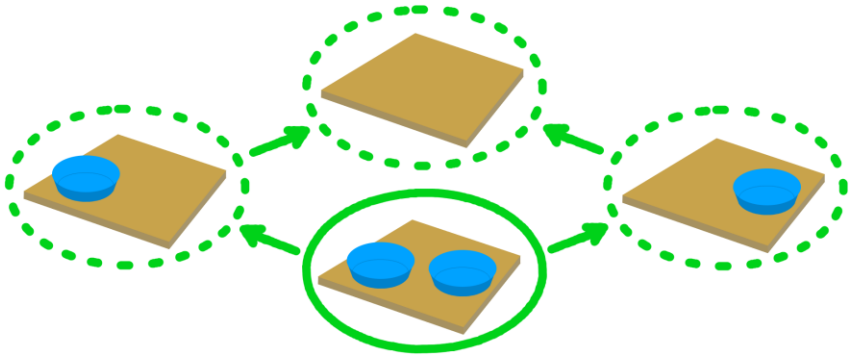
Partial State Caching

- Full state contains values irrelevant for computation
- Use partial state $s^P \sqsubseteq s$ for caching, where s^P is a partial variable assignment
- Matches set of states $\{s' \in S : s^P \sqsubseteq s'\}$
- Needs module specific implementation as planner can't know what's relevant
  - Minimal variables are easy/straight-forward to determine from computation

SPATIAL COGNITION

Subsumption Caching

- Reuse information from different requests
- On request: Check if request is less constrained than some success s, o
- On miss: Remove cached states that are less constrained
- Analogous for failures



State s^- is less constrained than s for operator o if

$$applicable(s, o) \Rightarrow applicable(s^-, o)$$

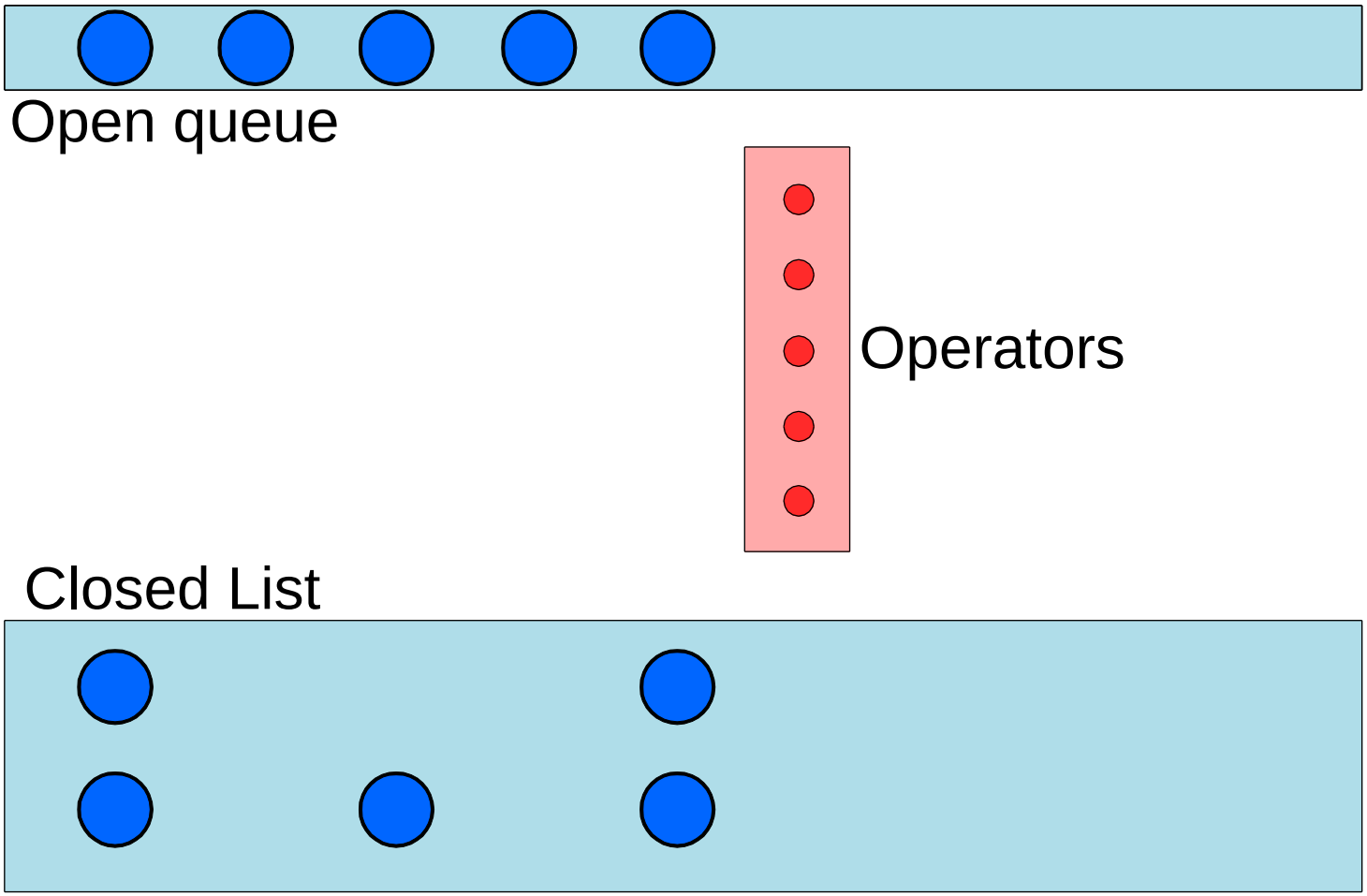
SPATIAL COGNITION

Global Caching

- Complementary to other methods
- Retain cache results between planner runs in persistent storage
- Requires to be able to store complete request for soundness

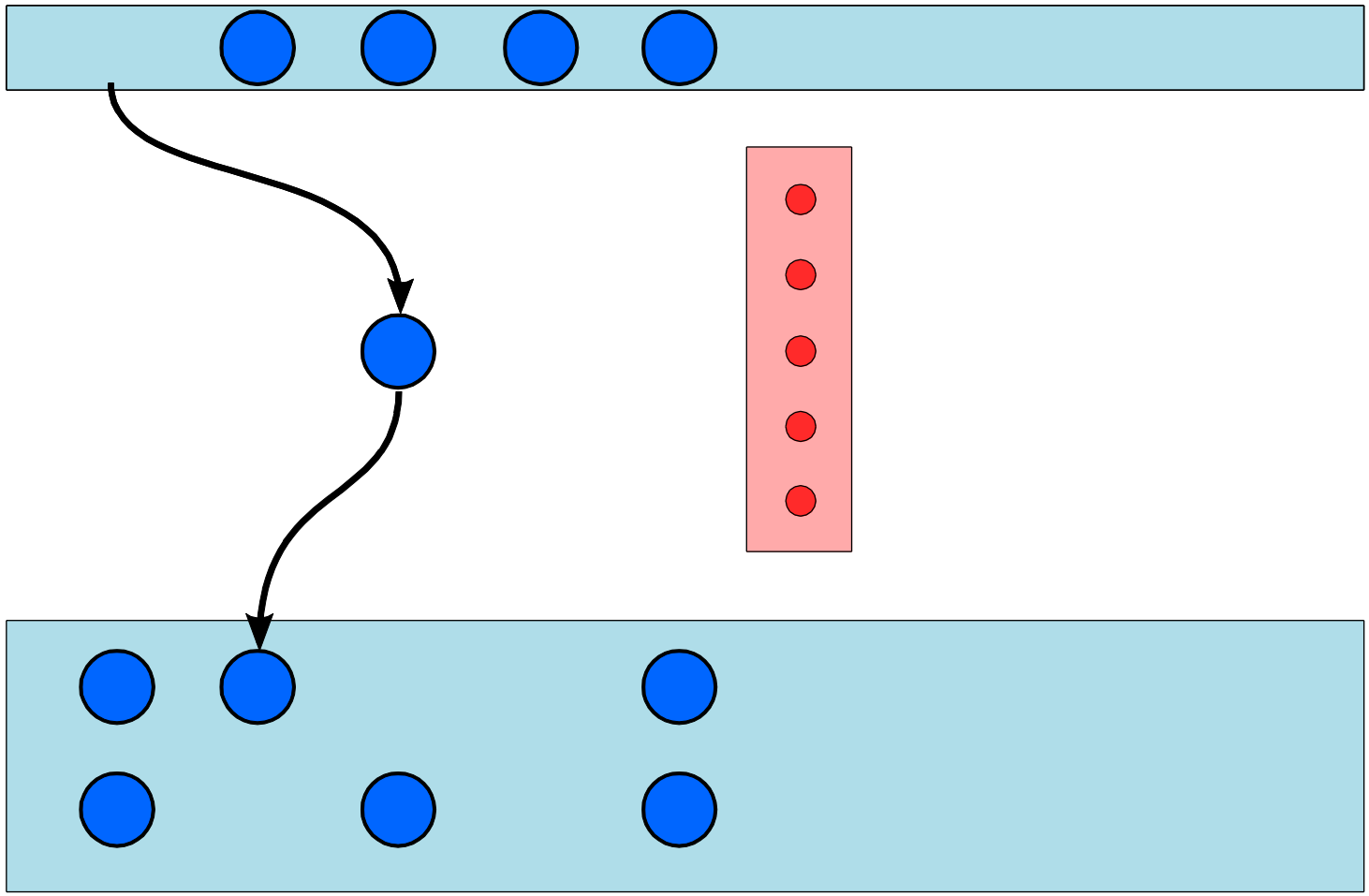
SPATIAL COGNITION

Lazy Module Evaluation

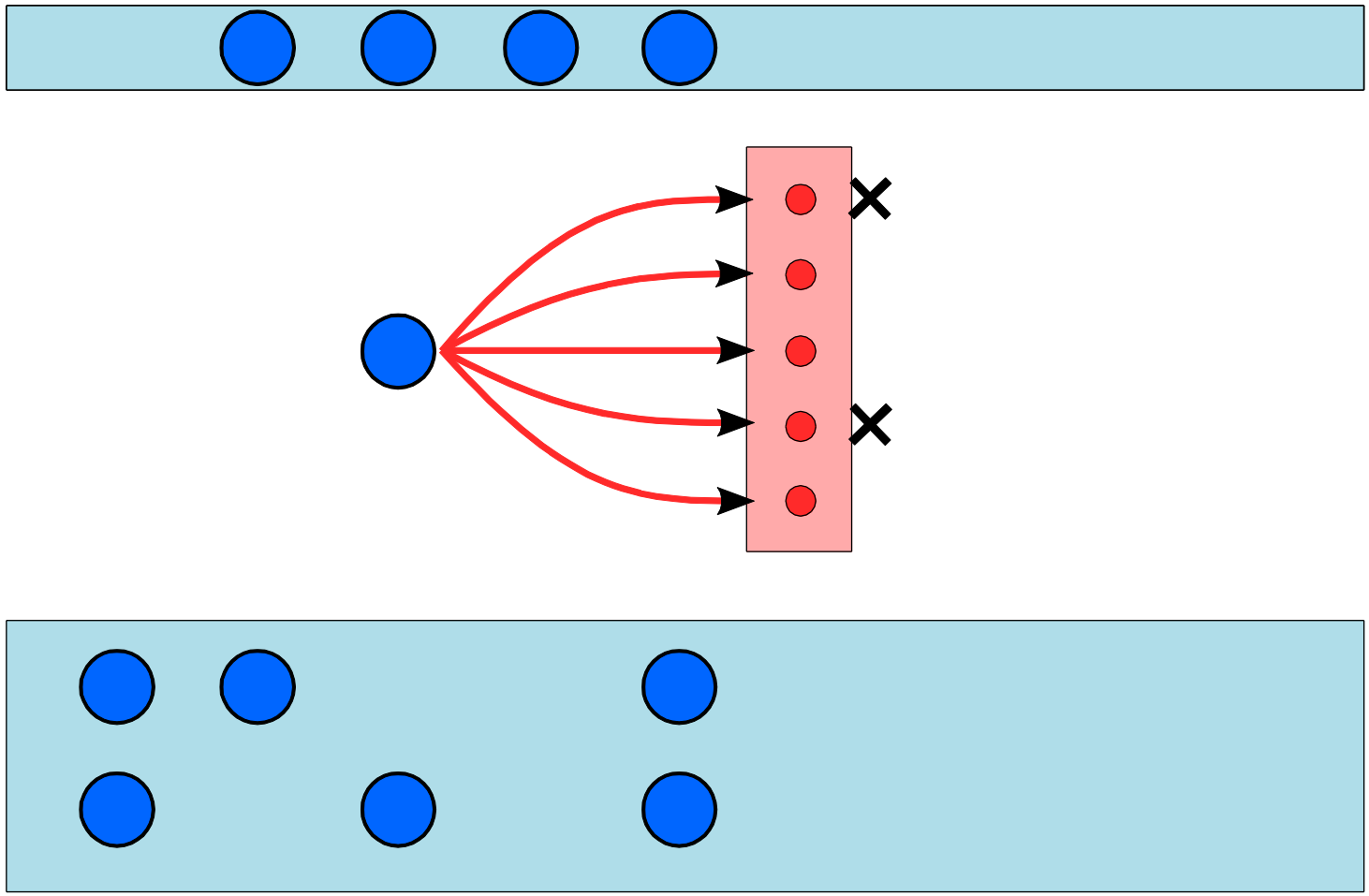


SPATIAL COGNITION

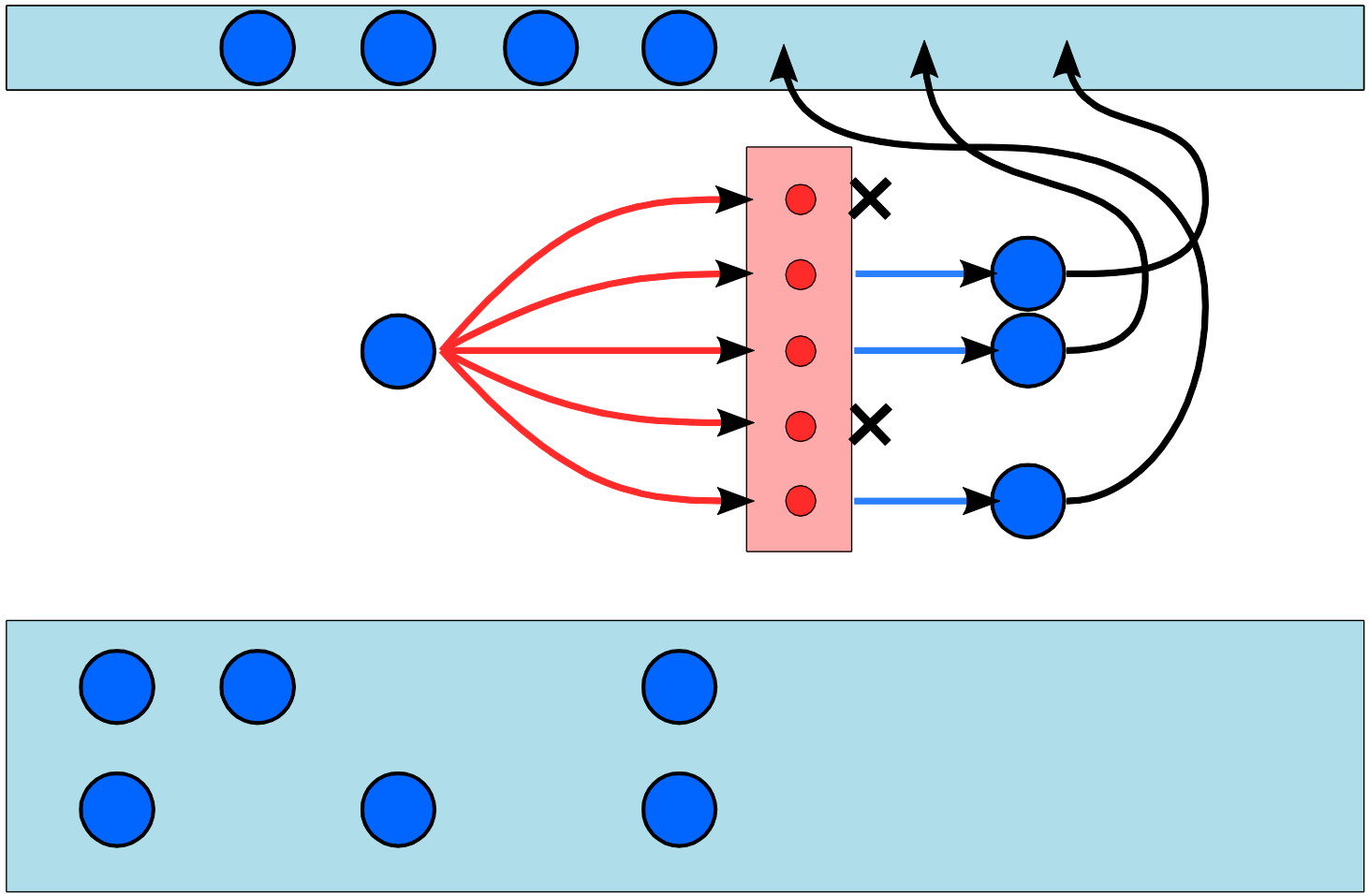
Fetch Next State



Applicability Check



Successor Generation



Lazy Evaluation

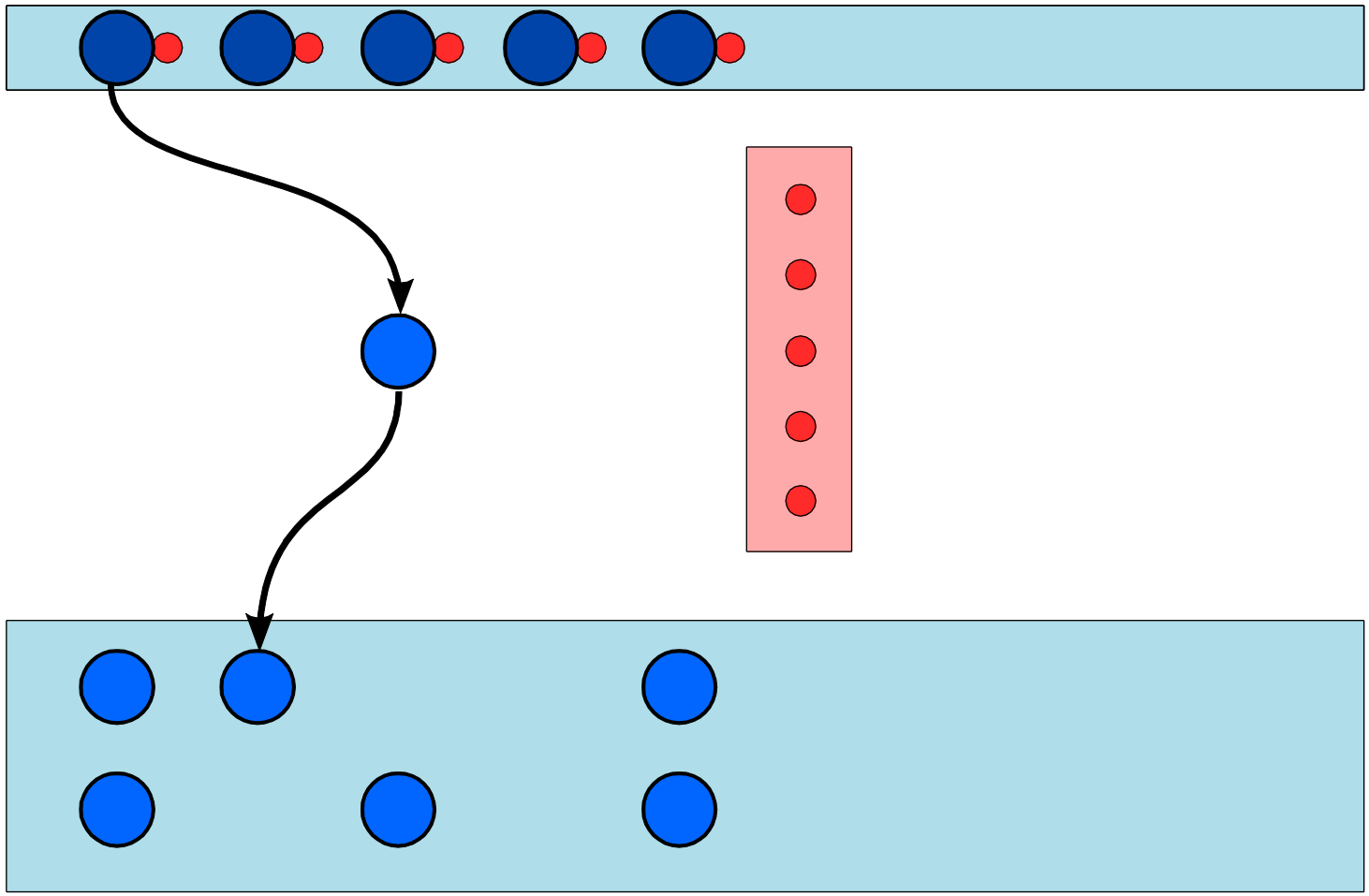
- Applicability checks slow
- Performed for open state – that might never be visited
- Replace by module relaxed applicability

For an operator o the operator o^+ is a relaxation if for all states s

$$applicable(s, o) \Rightarrow applicable(s, o^+)$$

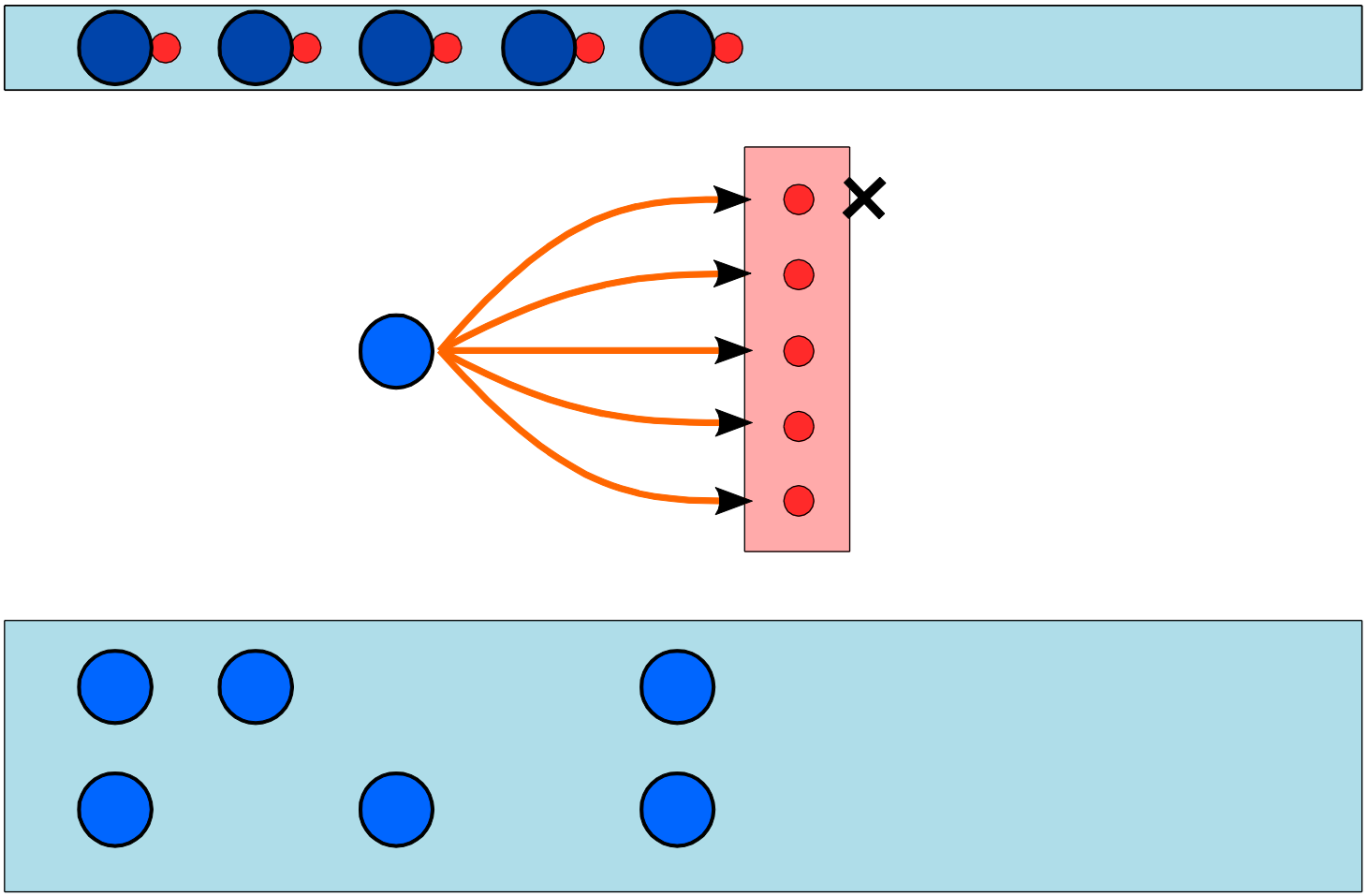
SPATIAL COGNITION

Lazy Evaluation

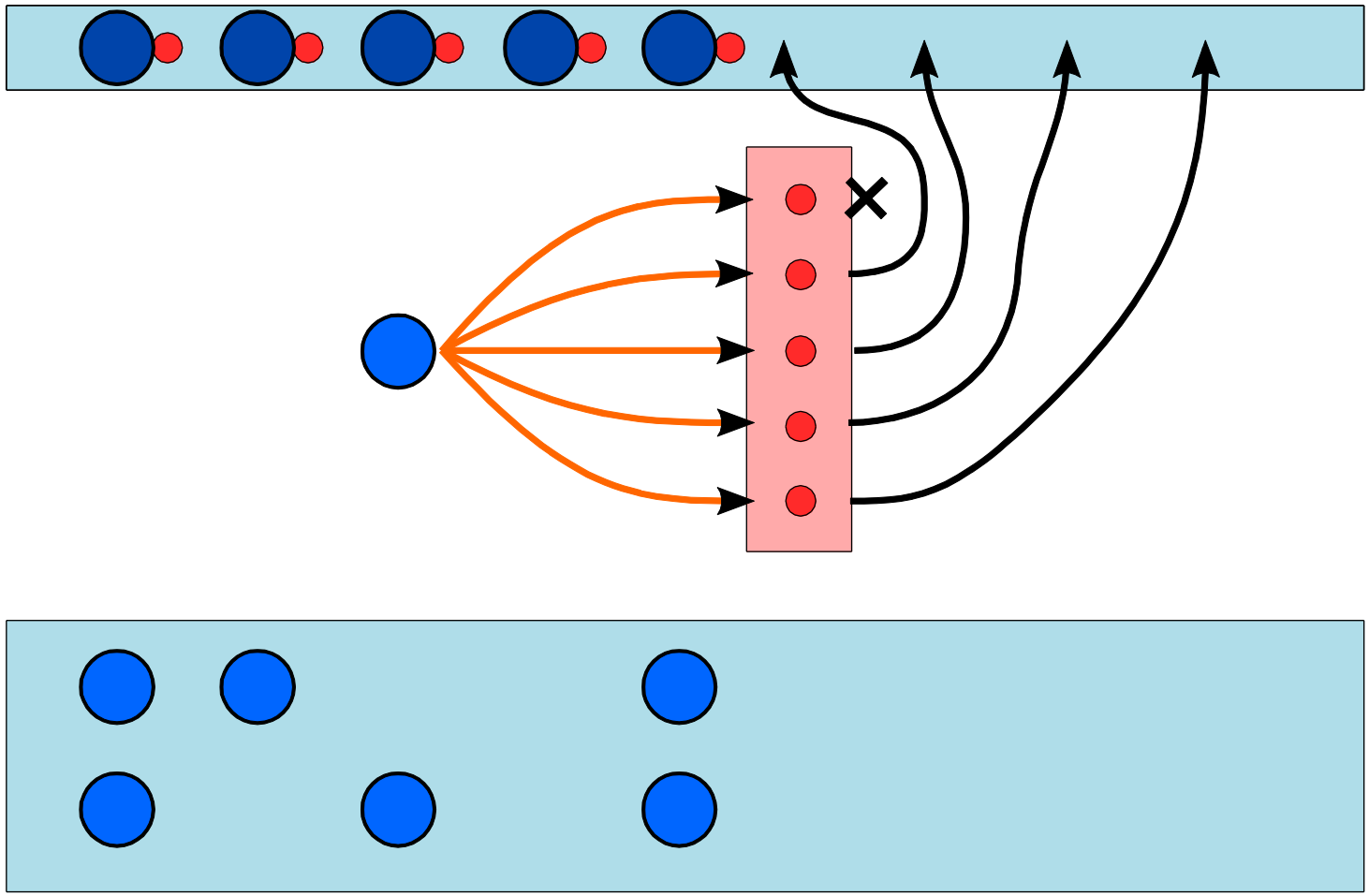


SPATIAL COGNITION

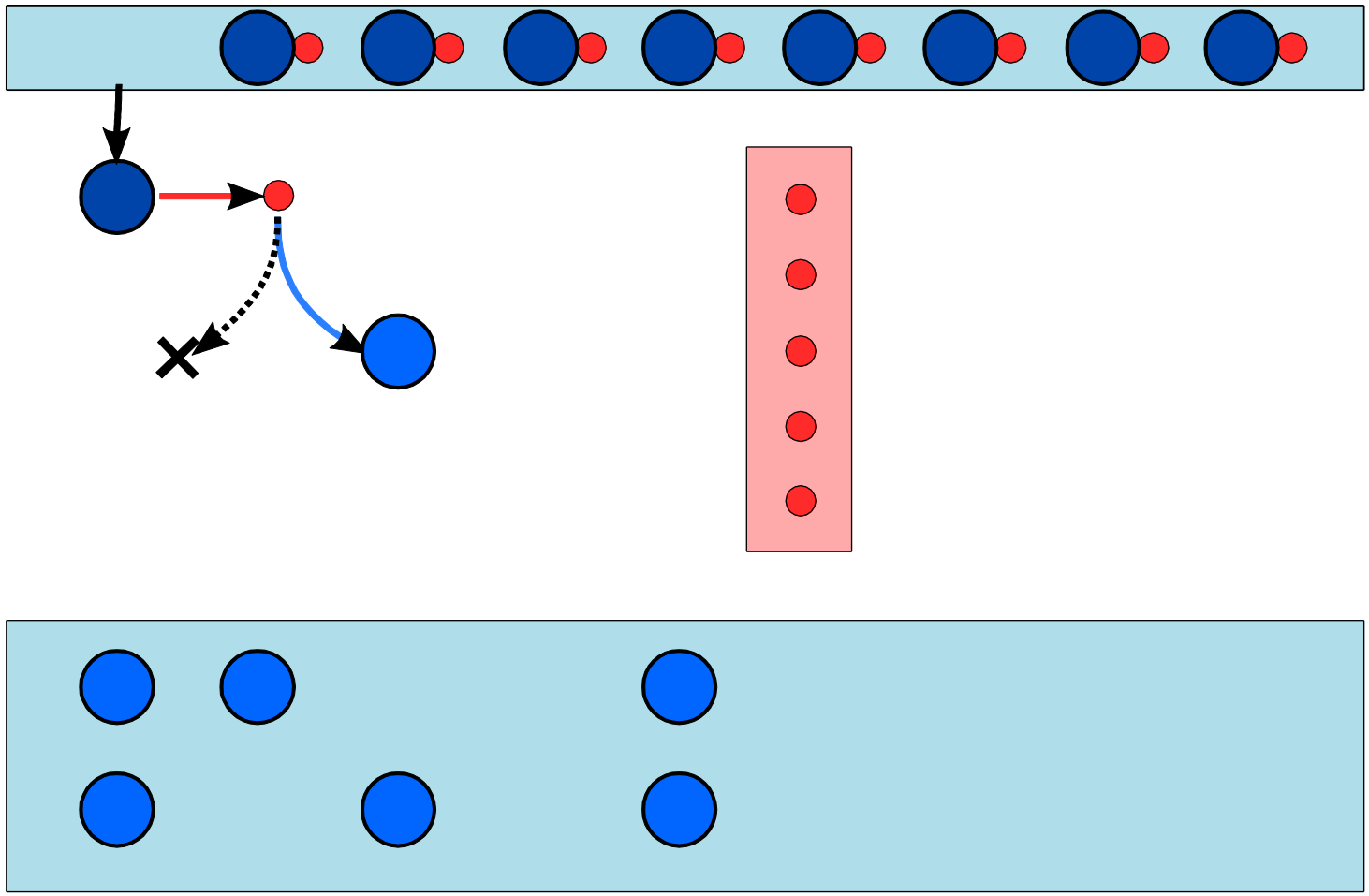
Relaxed Applicability Check



Successors: State + Operator

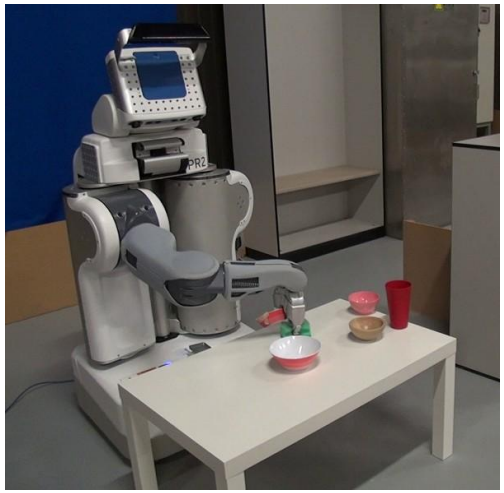


Fetch Next State + Applicability



Experiment Settings

- All objects to front table and wipe spots
- 1-5 objects, 2 configurations each
- Modules for navigation cost, wiping, putdown poses
- Anytime planning with 25% extra time
- Run each task to completion with replanning



Results – Eager/Lazy

Task	Objects	Total Planning Time [s]		Max Single Plan Time [s]		Planner Calls	Execution Time [s]	Actions
		Eager	Lazy	Eager	Lazy			
1	1	76.7	41.8	72.0	37.3	2	497.2	24
2	1	66.0	43.0	61.2	39.0	2	303.4	21
3	2	57.4	42.3	47.9	32.2	3	807.8	38
4	2	106.2	71.2	73.2	42.2	4	631.8	40
5	3	221.0	158.1	112.6	76.6	4	823.8	46
6	3	124.0	99.8	42.4	27.7	18	1630.9	94
7	4	289.6	220.8	203.1	153.5	10	1226.1	63
8	4	120.9	99.8	57.8	56.4	4	1019.3	55
9	5	686.2	505.3	263.7	220.1	6	1651.7	82
10	5	350.2	255.5	281.3	211.0	3	1195.0	56

- Total planning time scales with execution time
  - Fast enough for practical systems
- Lazy module evaluation is faster in comparison to eager

SPATIAL COGNITION

Results – Caching Putdown

Task	Number of Requests	No Caching Time [s]	Full State Misses	Full State Time [s]	Full State (G) Misses	Full State (G) Time [s]	Partial State Misses	Partial State Time [s]	Partial State (G) Misses	Partial State (G) Time [s]	Subsumption Misses	Subsumption Time [s]	Subsumption (G) Misses	Subsumption (G) Time [s]
1	98	60.5	60	36.7	60	36.7	8	4.5	8	4.5	8	4.5	8	4.5
2	140	101.0	70	49.7	70	49.7	5	3.0	5	3.0	5	3.0	5	3.0
3	347	170.4	224	110.5	224	110.5	22	11.3	17	8.9	20	10.5	15	8.0
4	1124	898.7	402	318.9	402	318.9	37	27.0	32	23.0	37	27.0	32	23.0
5	7488	4701.7	1954	1184.1	1954	1184.1	70	46.4	62	42.2	68	45.3	62	42.2
6	2559	1422.1	1568	862.4	1568	862.4	200	121.7	68	46.8	178	111.7	95	71.1
7	5234	3411.2	1880	1285.5	1848	1259.7	175	140.2	90	69.1	159	126.6	74	55.5
8	1167	790.9	540	358.3	540	358.3	66	40.3	66	40.3	66	40.3	66	40.3
9	11823	15907.6	3980	5301.0	3980	5301.0	247	472.1	228	413.1	203	393.7	190	340.9
10	6900	4230.3	1844	1112.8	1844	1112.8	63	38.8	63	38.8	62	37.6	62	37.6

- No caching/full state caching impractical
- Largest improvement full state vs. partial state caching
- Subsumption caching effective mainly for larger tasks

- (Global caching worked well for navigation)

SPATIAL COGNITION

Example Task: 3 Objects (15x)

SPATIAL COGNITION

Summary/Discussion

- Geometric computations are expensive in any symbolic reasoning tool
- Avoid them whenever possible
- Caching techniques work
  - Partial state is the intuitive implementation
  - Subsumption caching theoretically better, but needs non-trivial constraints
- Lazy module evaluation is faster

SPATIAL COGNITION

Lazy Evaluation and Subsumption Caching for Search-Based Integrated Task and Motion Planning

Christian Dornhege, Andreas Hertle, Bernhard Nebel

{dornhege,hertle,nebel}@informatik.uni-freiburg.de

Foundations of Artificial Intelligence

University of Freiburg

SPATIAL COGNITION